UIT Development: How to Access HPC Resources Using the UIT Web Service

Wes Monceaux, Keith Rappold, and Patti Duett
Information Technology Laboratory
Scotty Swillie and Robert S. Maier
Major Shared Resource Center
U.S. Army Engineer Research and Development Center
Vicksburg, MS
Weston.P.Monceaux@erdc.usace.army.mil
Keith.N.Rappold@erdc.usace.army.mil
Patti.S.Duett@erdc.usace.army.mil
Scotty.Swillie@erdc.usace.army.mil
Robert.S.Maier@erdc.usace.army.mil

Abstract

The User Interface Toolkit (UIT) is a Web service that provides an application programming interface (API) for accessing and manipulating high performance computing (HPC) resources. Developers can use this API to create custom software that can make use of HPC resources as desired. This presentation will demonstrate how to make use of existing development tools to access the UIT Web service API.

One of the objectives of the UIT is to provide a way for developers to more easily create software that accesses HPC resources. By making it easier to create HPC-aware desktop applications, many users would be able to benefit from the resources the HPC has to offer. This goal is accomplished through the Web service API. The goal of this presentation is to give interested developers a jumpstart into the usage of the API.

Developers will be shown how to access the API using two predominant software development environments. The first will use Microsoft's Visual Studio to access the API in the .NET development environment. The second will use Apache's Axis Web service libraries to demonstrate API usage in a Java development environment. Other languages and available tools will be mentioned.

The anticipated result is that many developers will feel more comfortable with using the UIT API to construct custom software that accesses HPC resources. At the very least, developers should have a starting point with what tools will be required and how they can expect to use them.

Providing an API for accessing HPC resources is useless if developers do not understand how to use it. The investment in the UIT API has a higher return rate as the number of developers that use it rise. It only makes sense to demonstrate to developers how to make use of the UIT API.

1. Introduction

The User Interface Toolkit (UIT) is a Web service that provides an application programming interface (API) for accessing and manipulating high performance computing (HPC) resources. Application developers with a need to utilize the computing power of the HPC systems can use this API to create custom software that can make use of HPC resources as desired. This paper will discuss how to make use of existing development tools to access the UIT Web service API.

In this paper, developers will be shown how to access the API using the two currently dominant software development environments for doing web service client development. The first will use Microsoft's Visual Studio to access the API in the .NET development environment. The second will use Apache's Axis Web service libraries to demonstrate API usage in a Java development environment. SOAP tools available for other programming languages and environments will be mentioned.

2. Objectives

One of the objectives of the UIT is to provide a way for developers to more easily create software that accesses HPC resources. By making it easier to create HPC-aware desktop applications, many users would be able to benefit from the resources the HPC has to offer. This goal is accomplished through the UIT web service API. The goal of this paper is to give interested developers a jumpstart into the usage of the API.

3. How Web Services work

A web service is a software system that allows communication between machines over a network. The service exposes an interface defined in a format such as WSDL which describes how to interact with the service. This communication often consists of XML and is normally sent by HTTP. The messages sent are often enclosed in a SOAP envelope which is a standard protocol for sending XML over HTTP.

Because web services are built on open standards, it allows for interoperability between different programming languages. [1]

4. Access with dotNET

This paper will attempt to show how to use Microsoft dotNET to create an application that will make use of the UIT API. The application will call the authenticate method within the API to get a token. This token will then be used to call getPublicHosts method to return a list of available public hosts on the MSRC network.

4.1 Prerequisites

Visual Studio is the primary requirement for developing applications that run within the .NET runtime environment. Information about visual studio can be found at http://msdn.microsoft.com/vstudio.

Additionally, SSL certificates are required to allow for SSL encryption between the .NET client and the web server. Visual Studio accesses the same root certificate keystore as Internet Explorer. For this walkthrough, install the DoD root certificates within IE by running "InstallRoot.exe". Instructions for this can be found at http://www.onr.navy.mil/resources/instructions.asp.

4.2 Creating the Application

This paper will demonstrate how to create a dotNET application to utilize UIT. For the purposes of demonstration, an application will be created that simply authenticates using kerberos and then lists the public hosts available for use.

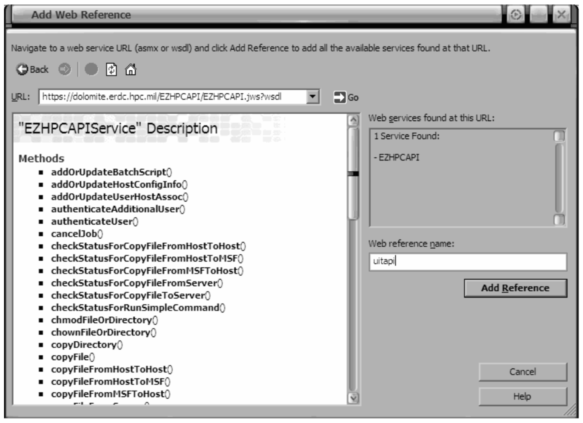
4.2.1 Creating the Project

First, create a new project for your application. Choose File, New Project. For this application, select "Visual C# project" from the project type list and then "Windows Application" from the template list. Give your project a name in the box below. For this project, type "UITTestApp".

4.2.2 Adding a Web Reference

A blank GUI form will appear on the left side of the Visual Studio IDE. On the right side, there will be a "solution explorer". Right click on "references", "add web reference" and you will be presented with a "add web reference" window.

In the URL textbox, type the url reference to the UIT web service. For this example, use https://www.uit.hpc.mil/UITAPI/UITAPI.jws?wsdl. Click "go" and a window with the API methods listed will be shown. Give the web reference a name in the textbox on the bottom right ("uitapi") and click "add reference". In the "solution explorer", there should now be a "web reference" with "uitapi" shown below it.



Add Web Reference

Navigate to a web service URL (asmx or wsdl) and click Add Reference to add all the available services found at that URL.

URL: https://dolomite.erdc.hpc.mil/EZHPCAPI/EZHPCAPI.jws?wsdl

"EZHPCAPIService" Description

Methods

- addOrUpdateBatchScript()
- addOrUpdateHostConfigInfo()
- addOrUpdateUserHostAssoc()
- authenticateAdditionalUser()
- authenticateUser()
- cancelJob()
- checkStatusForCopyFileFromHostToHost()
- checkStatusForCopyFileFromHostToMSF()
- checkStatusForCopyFileFromMSFToHost()
- checkStatusForCopyFileFromServer()
- checkStatusForCopyFileToServer()
- checkStatusForRunSimpleCommand()
- chmodFileOrDirectory()
- chownFileOrDirectory()
- copyDirectory()
- copyFile()
- copyFileFromHostToHost()
- copyFileFromHostToMSF()
- copyFileFromMSFToHost()

Web services found at this URL:

1 Service Found:

- EZHPCAPI

Web reference name: uitapi

Figure 1. Adding a web reference in dotNET.

4.2.3 Creating a Form

Create a simple form to accept kerberos credential information, authenticate, and make the api call to get a list of hosts available. Click on the "toolbox" toolbar on the left side and drag 4 textboxes to the form. Drag a label beside the first , second, and third textboxes.

Click on the first label and change its text property in the properties window (located in the bottom right corner of the Visual Studio IDE) to "Principal". Change the following two labels to "Password" and "Passcode" respectively.

For the fouth textbox, change the multi-line property to true. For the Password textbox, change its PasswordChar property to "*".

Drag a button to the bottom of the form and change its text property to "Login". When complete, the form should look like Figure 2.

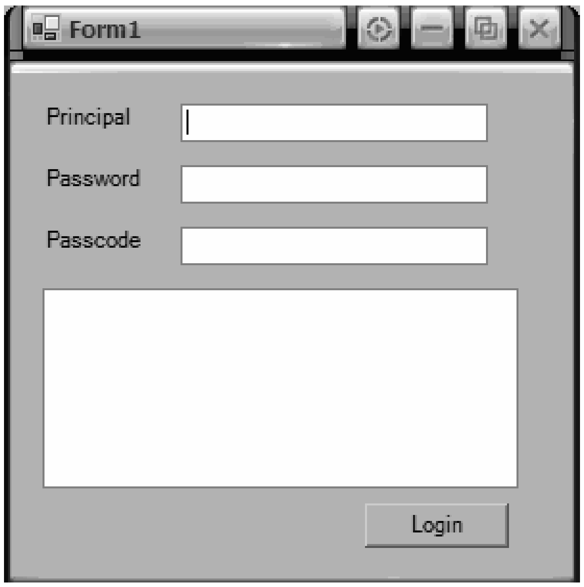


Figure 2. Example login form.

4.2.4 Calling the UIT API

Double click the login button which will bring up the coding screen. The cursor will already be within the coding section for the button_Click action. In this section, add the following lines of code to create a new instance of web service we added the reference for at the beginning.

```
uitapi.EZHPCAPIService uitws = new UIT-
TestApp.uitapi.EZHPCAPIService();
```

The authenticate method within UIT requires two strings. The first is an XML credentials string that contains the information from the form. The second is the authentication method which will be acquired from an additional method call. First the XML credentials string can be constructed with the following code.

```
string xmlcreds = "<credentials><username>"+
textBox1.Text + "</username><password>"+
textBox2.Text + "</password><passcode>"+
textBox3.Text + "</passcode></credentials>";
```

The second string is the authentication method and can be acquired from the getRequiredTransportMethods api call. This call will return a string array of authentication methods. The first method in the array should me the Kerberos authentication method and is the one we want.

```
string[] authMethods =
ezws.getRequiredTransportMethods();
string msrcAuth = authMethods[0];
```

Call the authentication method which will return a token to be used with additional api calls.

```
String token =
ezws.authenticateUser(xmlcreds, msrcAuth);
if ((token == null) || (token. Equals("")))
Application.Exit();
```

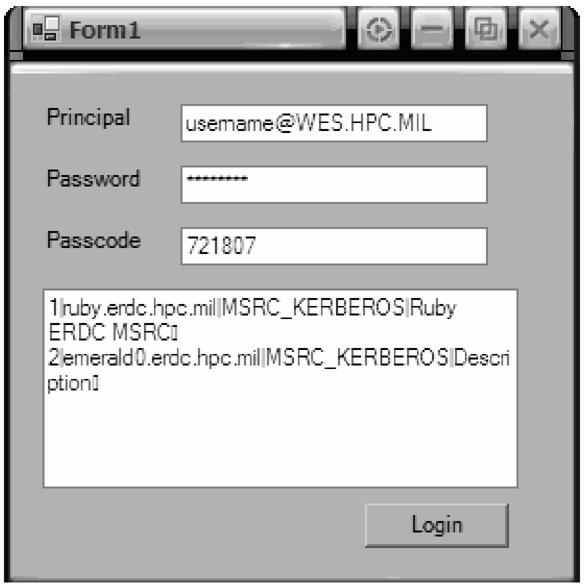
Call the getPublicHosts API method to get a string array listing of the hosts available.

```
string[] hosts =
uitws.getPublicHosts(token);
```

Display the public hosts in the multiline textbox.

```
for (int i=0; i < hosts.Length; i++)
    textBox4.AppendText(hosts[i]+"\n");
```

Run the application by pressing the F5 button to start the application in debug mode. The results of the run should look like Figure 3.



Principal: username@WES.HPC.MIL
Password: ********
Passcode: 721807

```
1|ruby.erdc.hpc.mil|MSRC_KERBEROS|Ruby
ERDC MSRC
2|emerald0.erdc.hpc.mil|MSRC_KERBEROS|Descri
ption
```

Figure 3. List of public hosts.

5. Access with Axis

This paper will now show how to use java to create an application that will make use of the UIT API. The application will again call the authenticate method within the API to get a token. This token will then be used to call getPublicHosts method to return a list of available public hosts on the MSRC network.

5.1 Prerequisites

The four requirements for developing this application are as follows:

- A current version of the java jdk (http://java.sun.com/javase/downloads/index.html). Version 5.0 JDK at the time of this publication.
- A current version of apache axis (http://ws.apache.org/axis/). Version 1.3 at the time of this publication.
- An editor to write code in. For this publication, we will use netbeans IDE (http://www.netbeans.org/)
- Windows XP Professional

5.2 Creating the application

This paper will now demonstrate how to create a java application to utilize UIT. Again an application will be created that simply authenticates using Kerberos and then lists the public hosts available for use.

5.2.1 Enable SSL

As with the dotNET example, an SSL certificate will be needed to enable java to communicate with the server over SSL.

Go to the UIT website (www.uit.hpc.mil) and download the SSL cerificate to the directory where the Java cacerts file is located (ex: C:\j2sdk1.4.2_04\jre\lib\security).

Import the ssl certificate into the cacerts keystore using keytool.

```
C:\j2sdk1.4.2_04\jre\lib\security> keytool -
import -trustcacerts
-alias uitcert -keystore cacerts
-file dod.cer
```

When prompted for keystore password, type changeit (unless the password for the keystore has already been changed). When prompted for "trust this certificate", type yes.

5.2.2 Use Apache Axis to Generate Stubs

Open a browser to the UIT WSDL file (https://www.uit.hpc.mil/UITAPI/UITAPI.jws?wsdl).

Save the page as a .wsdl file. Choose File, save as "UITAPI.wsdl" and save it to the lib directory within apache axis.

Run WSDL2Java to create java stubs that will interact with the web service.

```
java -classpath
"axis-ant.jar:
axis-schema.jar:
axis.jar:
commons-discovery-0.2.jar:
commons-logging-1.0.4.jar:
jaxrpc.jar:log4j-1.2.8.jar:
log4j.properties:
saaj.jar:
wsdl4j-1.5.1.jar"
org.apache.axis.wsdl.WSDL2Java
UITAPI.wsdl -p uittest
```

This will create a directory or package named uittest that contains the java stubs necessary to interact with UIT.

5.2.3 Create a Project

Start a new project for this demonstration. From the Netbeans IDE, select: File, New Project. Click next.

On the new project page, select general from the categories list and java application from the projects list. Click next.

On the name and location page, give your project a name. UITDemo was chosen as the name for this demo. Additionally, we gave our main class a name of uit_demo_main. Click finish.

5.2.4 Add Libraries to the Project

Right click on the UITDemo project and select properties. Choose sources, add folder and choose the directory where the Axis generated stubs are located.

Create a library that points to the axis jar files. Go to tools, library manager and click on new library and give it a name of axis. Now select add jar/folder and choose the directory where the axis jars are located.

Select your project UITDemo and go to file, project properties. Click on libraries, add library and choose axis. This library is now part of the project.

5.2.5 Calling the UIT API

Use your generated stubs to create a new instance of the API service.

```
uittest.EZHPCAPIServiceLocator
  locator =
  new uittest.EZHPCAPIServiceLocator();
uittest.EZHPCAPI_PortType api = null;

try
{
  api = locator.getEZHPCAPI();
}
catch(Exception e){}
```

Prompt the user for the principal password and passcode required for authentication.

```
java.io.BufferedReader br = new
java.io.BufferedReader(new
java.io.InputStreamReader(System.in));
    String principal = "";
    String password = "";
    String passcode = "";
    try {
       System.out.println("Enter principal:
username@WES.HPC.MIL: ");
       principal = br.readLine();
       System.out.println("Enter password:
");
       password = br.readLine();
       System.out.println("Enter passcode:
");
       passcode = br.readLine();
    } catch (Exception e) {}
```

```
```

The authenticate method within UIT requires two strings. The first is an XML credentials string that contains the information from the form. The second is the authentication method which will be acquired from an additional method call. First the XML credentials string can be constructed with the following code. The principal, password, and passcode strings should be properly XML encoded (replace ampersands (&) with the XML encoded "`&amp;`" value, etc.).

```
String credentials = "<creden-
tials><username>"+ principal +
"</username><password>"+ password +
"</password><passcode>"+ passcode +
"</passcode></credentials>";
```

Secondly, one must call the method getRequiredTransportMethods to get a list of available transport methods for authentication. Kerberos transport is the only one available at the time of this publication.

```
String[] tmp =
api.getRequiredTransportMethods();
String defAuthMethod = tmp[0];
```

Call authenticateUser to get back a token to use with other method calls.

```
token = api.authenticateUser(credentials,
defAuthMethod);
```

Call getPublicHosts to get a list of available public hosts.

```
if (token != null && token != ""){
  String[] hosts =
api.getPublicHosts(token);
  for (int i=0; i < hosts.length; i++)
         System.out.println(hosts[i]+"\n");
}
```

This should generate a list of public hosts which are available for use within the UIT.

6. Other tools

Below is a list of references to tools which can be used with other programming languages:

- nuSOAP. SOAP Toolkit for PHP. http://sourceforge.net/projects/nusoap/
- Axis C++. C++ libraries for SOAP. http://ws.apache.org/axis/cpp/index.html
- SOAP::lite. Perl libraries for SOAP. http://soaplite.com/
- Python Web Services. A page with links for interacting with web services in perl. http://pywebsvcs.sourceforge.net/

7. Conclusion

Once a developer has the proper tools configured for working with SOAP web services, it is easy to see that making basic use of the UIT API is not that difficult. Hopefully, this paper has given those developers interested in using the UIT API a quick primer on how to get started.

8. Terms

- SOAP (Simple Object Access Protocol). A protocol for exchanging XML-based messages over a computer network, normally using HTTP.[2]
- WSDL (Web Services Description Language). An XML format published for describing Web services.[3]
- XML (Extensible Markup Language). A way of describing data and an XML file can contain the data too, as in a database.[4]
- SSL (Secure Sockets Layer). Cryptographic protocols which provide secure communications on the Internet.[5]

9. References

1. "Web Service." http://en.wikipedia.org/wiki/Web_service, May 2006
2. "SOAP." http://en.wikipedia.org/wiki/SOAP, May 2006
3. "Web Services Description Language." http://en.wikipedia.org/wiki/Web_Services_Description_Language, May 2006
4. "XML" http://en.wikipedia.org/wiki/XML, May 2006
5. "Transport Layer Security" http://en.wikipedia.org/wiki/Secure_Sockets_Layer, May 2006